Description of video coding technology proposal

SK telecom

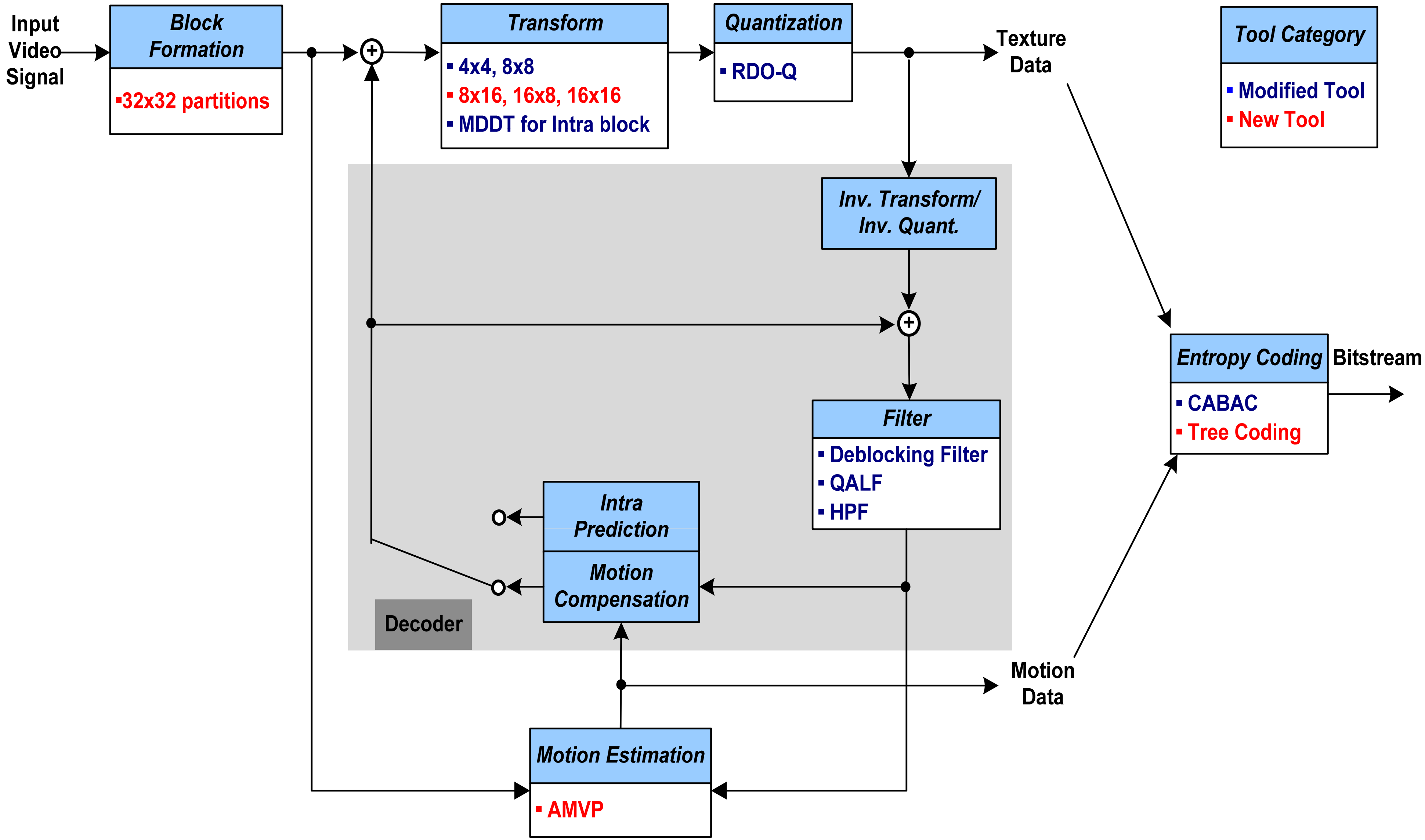
Sejong University

Sungkyunkwan University

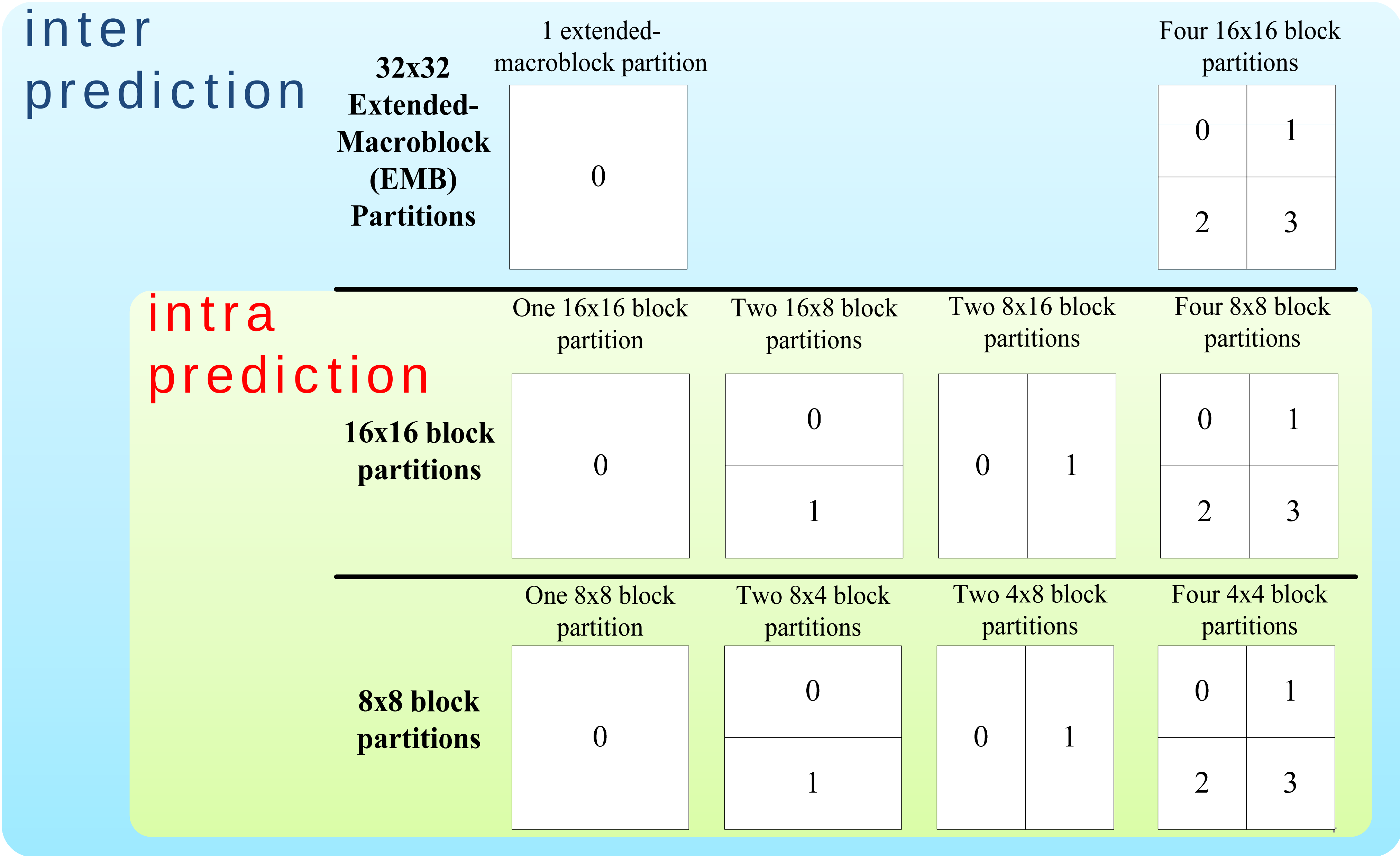
Summary

- This contribution presents video coding using extended-macroblock
  - Modified JM15.2 + MDDT+QALF+HPF
  - New Features
    - ✓32x32 macroblock (EMB) and new block partitions
      - Inter Mode: 32x32, 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4
      - Intra Mode: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4
    - ✓Adaptive Motion Vector Precision(AMVP)
      - ½ pel, ¼ pel, 1/8 pel
    - ✓Large Transform for Inter coding
      - 16x8, 8x16, 16x16
    - ✓Tree Coding for partition type information

Block diagram of the encoder



EMB and partitions



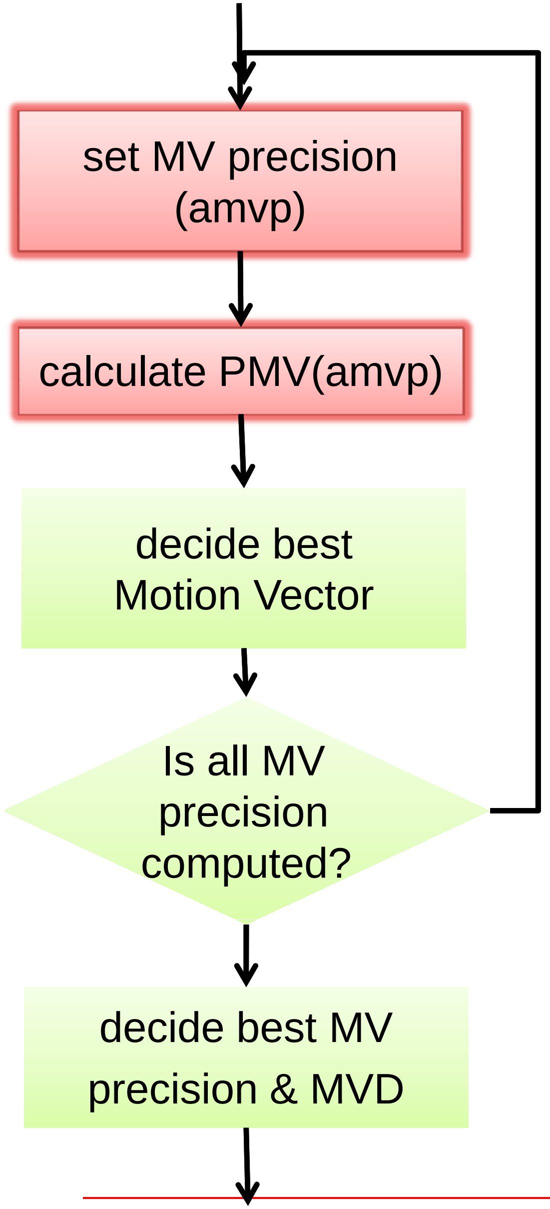
Motion Representation

- Inter Coding Mode
  - 32x32 SKIP, 16x16 SKIP
  - 32x32 Direct, 16x16 Direct, 8x8 Direct
  - AMVP motion and residual data coding

- Adaptive Motion Vector Precision(AMVP)
  - MV precision is adaptively selected among 1/2-pel, 1/4-pel, and 1/8-pel at EMB or 16x16 block
  - Interpolation filter

Step	Filter length	Impulse response	Interpolated position
1	6-tap	{1,-5,20,20,-5,1}/32	1/2 pel
2	2-tap	{16,16}/32	1/4 pel
3	2-tap	{16,16}/32	1/8 pel

PMV calculation in AMVP



❖ scaling neighboring MV

$$MV_x(amvp) = MV_x \times scaling\_factor$$

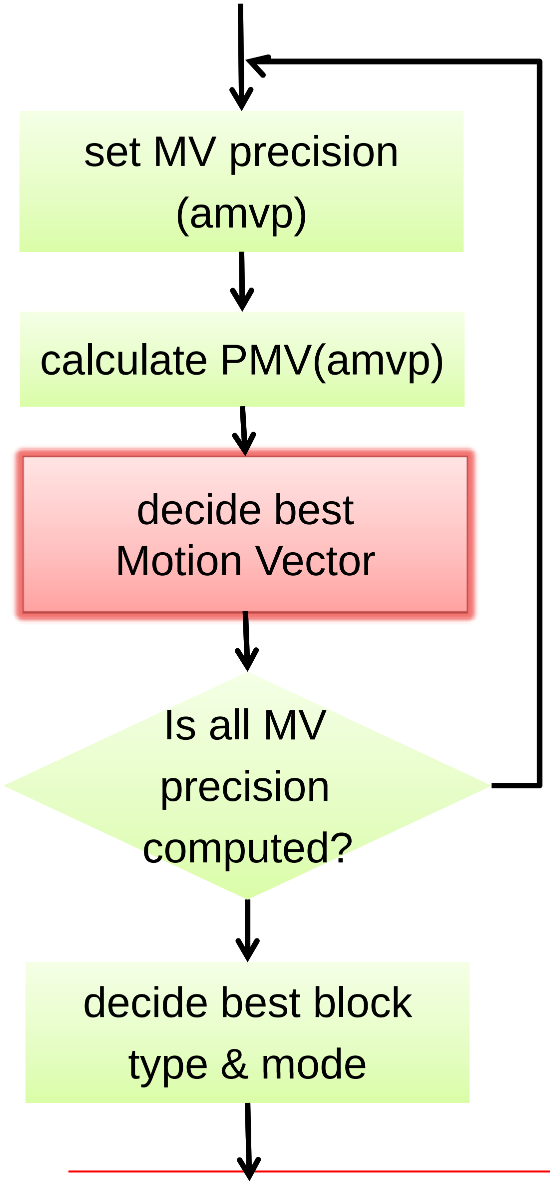
- $MV_x(amvp)$: scalied MV of neighboring block
- amvp: MV precision of current block
- Scailing_factor

AMVP Indicators	Neighboring PMV candidates		
	1/2	**1/4**	**1/8**
1/2	1	2	4
1/4	1/2	1	2
1/8	1/4	1/2	1

❖ PMV (Predictive MV)

$$PMV(amvp) = median(MV_a(amvp), MV_b(amvp), MV_c(amvp))$$

MV Estimation in AMVP



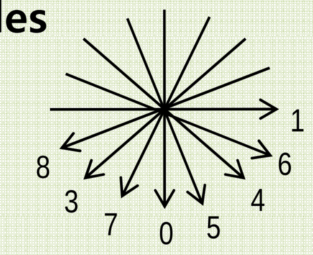
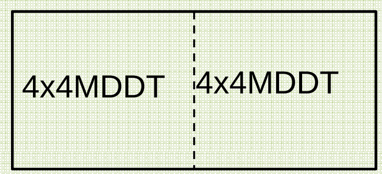
- $$Mcost = Dist_{motion} + \lambda_{motion} \times (R_{amvp} + R_{mvd})$$
  $$MVD(amvp) = MV(amvp) - PMV(amvp)$$
  - $Dist_{motion}$: distortion
  - λ_{motion} : a weighting factor depending on QP
  - R_{amvp} : the rate of MV precision
  - R_{mvd} : the rate of the MVD

AMVP indicator

- Represent the selected motion vector precision

MVD Accuracy	Codeword
1/4 pel	1
1/8 pel	00
1/2 pel	01

Intra Coding

Block size	**4x4, 8x8**	**16x16**	**16x8, 8x16 8x4, 4x8**	**Chroma**
available prediction mode	**9 modes** 	**4 modes** ✓0: Vertical ✓1: Horizontal ✓2: DC ✓3: Plane	**3 modes** ✓0: Vertical ✓1: Horizontal ✓2: DC	**4 modes** ✓0: DC ✓1: Horizontal ✓2: Vertical ✓3: Plane
encoding method of prediction mode	encode •most_probable_mode flag (1bit) •remaining_mode_selector (3bit)	encode prediction mode value (0~3)	encode •most_probable_mode flag (1bit) •remaining_mode_selector (1bit)	encode prediction mode value (0~3)
transform	•4x4 : 4x4 MDDT •8x8 : 8x8 MDDT	•16x16 MDDT	•4x8,8x4: 4x4MDDT •8x16,16x8: 8x8MDDT Intra8x4 	4x4 integer transform of H.264/AVC

DCT Transform size for inter block

	Luma					Chroma
Partition size	**4x4, 4x8, 8x4**	**8x8**	**16x8**	**8x16**	**16x16, 32x32**	**8x8**
available transform size	4x4	4x4 , 8x8	4x4, 8x8, 16x8	4x4, 8x8 8x16	4x4, 8x8, 16x16	4x4
transform size indicator	-	use Table 1	use Table 2			-

Table 1

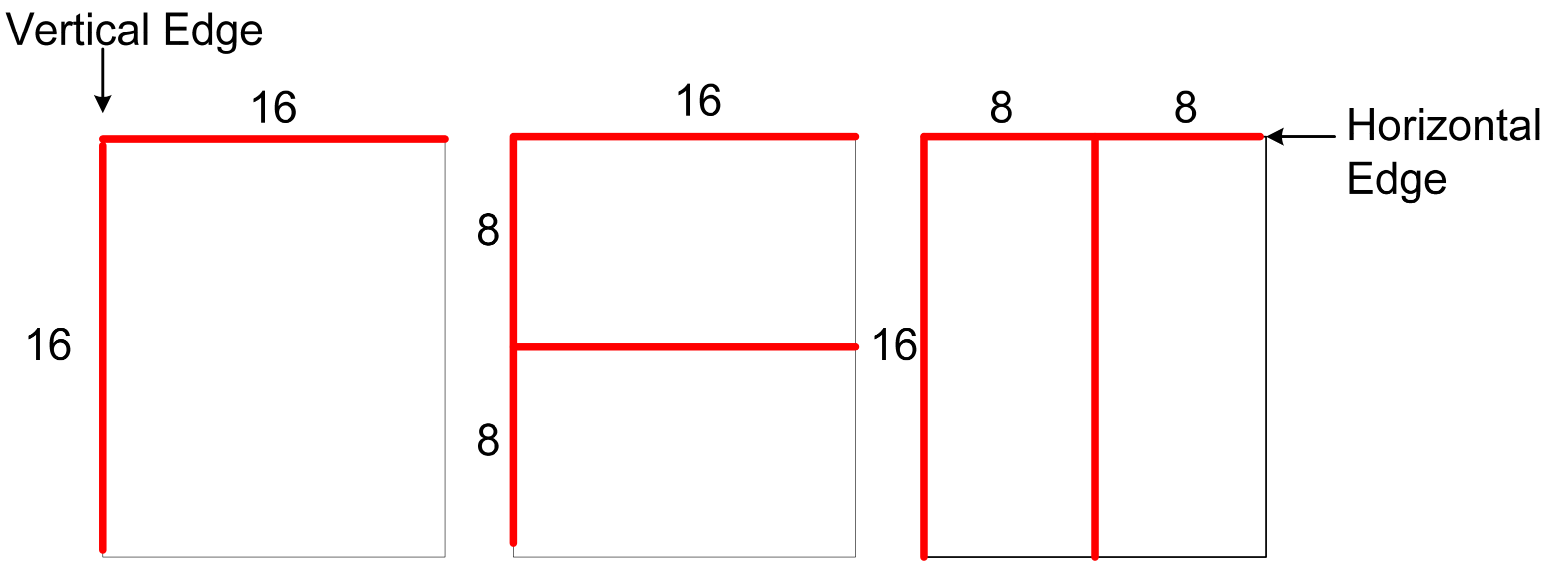
Transform_size	Bin string
4x4 transform	0
8x8 transform	1

Table 2

Transform_size	Bin string
4x4 transform	0
8x8 transform	10
8x16, 16x8 or 16x16 transform	11

Deblocking-loop filtering

- Transform boundaries are deblocking-filtered.
  - when large transform (16x16, 16x8, 8x16) is used, block boundary



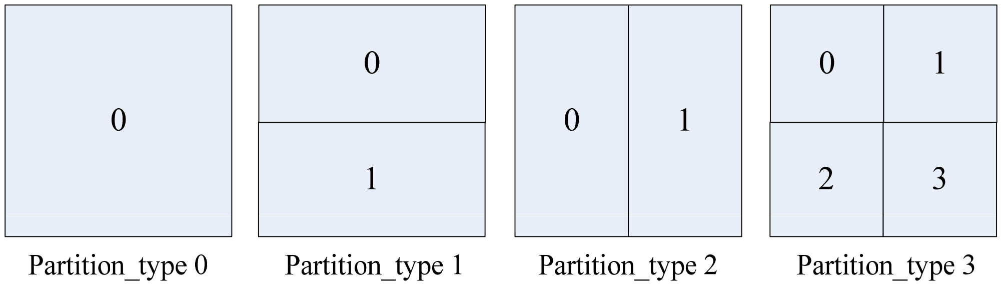
(a) block boundary when 16x16 transform is used

(b) block boundary when 16x8 transform is used

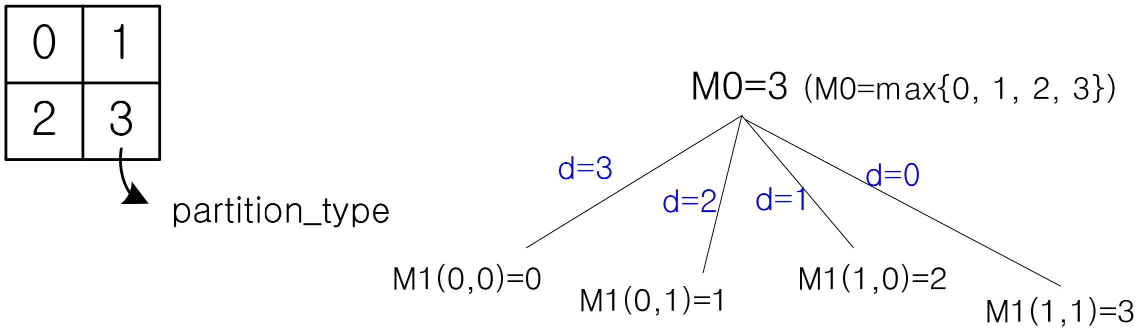
(c) block boundary when 8x16 transform is used

Tree Coding for partition_type[1]

❖partition_type value



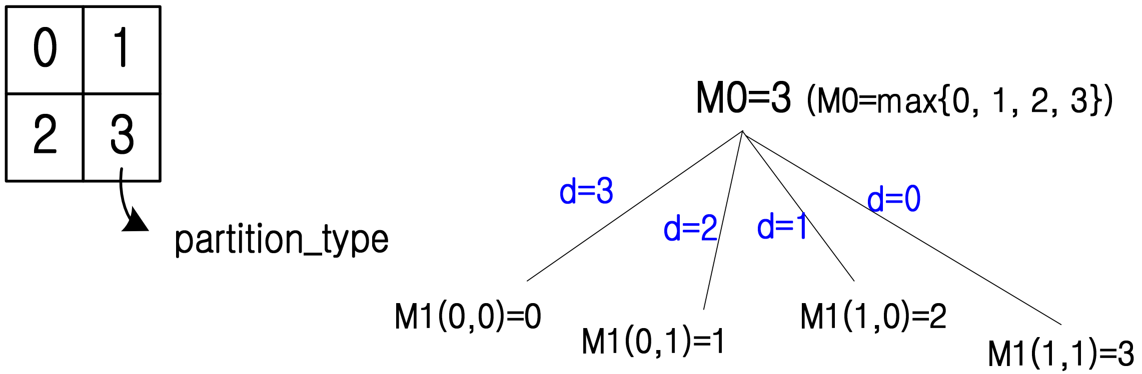
❖generate tree



– M1(0,0), M1(0,1), M1(1,0), M1(1,1): partition_type of 16x16 blocks
– M0 (root node): set to the maximum value of its children nodes

Tree Coding for partition_type[2]

JCTVC-A113



Node	Node value	d	Bin strings
M0	3	0	1
M1(0,0)	0	3	000
M1(0,1)	1	2	001
M1(1,0)	2	1	01
M1(1,1)	3	0	-

- encode **d** number of 0's followed by a 1
  - ✓d: difference value between **current node and** its parent node
  - ✓For the root, its parent node assumed maximum value of partition_type.
  - ✓If **d** is the biggest partition_type value, then do not encode 1

Coding for 8x8 partition_type

- encode bin string for 8x8 sub-block type value in the same way as H.264/AVC

Value of 8x8 partition_type	8x8 partition_type	Bin string
0	8x8	1
1	8x4	00
2	4x8	011
3	4x4	010

Test Conditions

Configuration	Constraint set 1	Constraint set 2
#.of reference frame	L0: 2 frames L1: 2 frames	4 frames
CABAC	Enabled	Enabled
8x8 Transform	Enabled	Enabled
RD Optimization	Enabled	Enabled
RDO-Q	Enabled	Enabled
Adaptive rounding	Disabled	Disabled
Fast motion estimation	Enabled (range 128x128)	Enabled (range 128x128)
RD Picture decision	**Disable**	**Disable**
Weighted Prediction	**Disable**	**Disable**
New offset	**Disable**	**Disable**
QALF	**Enable**	**Enable**
MDDT	**Enable**	**Enable**
HPF	**Enable**	**Enable**

Performance for Const. set1

JCTVC-A113

	Sequence	BD-PSNR	BD-RATE
ClassA	Traffic	0.638	-16.375
	People on street	0.799	-14.412
ClassB	**Kimono**	**1.027**	**-25.467**
	ParkScene	0.602	-15.411
	Cactus	0.482	-15.237
	Basketball Drive	**0.703**	**-20.521**
	BQTerrace	**0.501**	**-26.114**
ClassC	Basketball Drill	0.638	-14.428
	BQMall	0.963	-18.613
	PartyScence	0.673	-16.649
	RaceHorses	0.886	-19.637
ClassD	Basketball	0.707	-13.428
	BQSquare	**1.217**	**-29.300**
	Blowing Bubbles	0.654	-14.477
	RaceHorses	0.659	-12.294
	Average	0.739	-17.823

Performance for Const. set 2

	Sequence	BD-PSNR	BD-RATE
ClassB	**Kimono**	**0.842**	**-20.693**
	ParkScene	0.469	-12.857
	Cactus	0.357	-10.913
	Basketball Drive	**0.787**	**-21.341**
	BQTerrace	**0.657**	**-29.973**
ClassC	Basketball Drill	0.386	-9.404
	BQMall	0.552	-11.053
	PartyScence	0.377	-9.898
	RaceHorses	0.375	-9.313
ClassD	Basketball	0.385	-7.808
	BQSquare	0.520	-14.351
	Blowing Bubbles	0.141	-3.382
	RaceHorses	0.188	-3.731
ClassE	Vidyo 1	0.875	-19.740
	Vidyo 3	0.868	-19.348
	Vidyo 4	0.780	-18.975
	Average	0.535	-13.924

Complexity

- Conditions
  - executed on Intel Xeon two Quadcore CPUs 64 bit Windows 7 with 16G bytes memory
  - _ftime() function is used for measuring the computational complexity.
  - YUV output enabled and reference disabled.
- Average encoding time (compared with JM16.2)
  - Increase by 136.39% in constraint set 1
  - Increase by 199.73% in constraint set 2
- The decoding time of the proposed method is increased on the average by 199.01% in constraint set 1 and by 275.55% in constraint set 2 (compared with JM17.0)

Conclusions

- This contribution presents video coding using EMB
  - 32x32 macroblock (EMB) and new block partitions
  - Large Transform
  - Adaptive Motion Vector Precision(AMVP)
  - Tree Coding for partition type information
  - MDDT, RDO-Q, QALF

- Coding efficiency
  - const. set 1: average 17.8% bit reduction was achieved
  - const. set 2: average 13.9% bit reduction was achieved

- The following four areas are recommended to be further explored through core experiements
  - EMB partition structure
  - adaptive transform selection with larger transform sizes than 8x8
  - adaptive motion vector precision selection
  - efficient partition type information coding.